Bilingual movie sentiment analysis

Team 11

3 2023

Abstract

In this paper, we present a deep learning-based sentiment analysis system for Arabic and English movie reviews. To begin, we preprocess the reviews using stemming, stop word removal, and tokenization. On the preprocessed data, we train a Long Short-Term Memory (LSTM) neural network to forecast the sentiment of a specific review. Our algorithm successfully classified movie reviews as positive or negative with an accuracy of over 85% on the test set. We think that our research can advance the field of natural language processing and offer valuable information to producers, directors, and moviegoers.

1 Introduction

1.1 Motivation

The aim of this project is to develop a sentiment analysis model for movie reviews in both Arabic and English. Sentiment analysis is a key phase in the processing of natural language that can offer useful knowledge about people's attitudes and opinions toward a given subject. Due to the growing popularity of social media platforms and the huge amount of user-generated content, sentiment analysis has increased in importance as a field of study for businesses of all sizes. Most of the existing sentiment analysis algorithms were created for English text and might not work as well with texts in other languages. By creating a model that can precisely identify Arabic and English movie reviews as negative or positive, this study tackles this limitation. The suggested strategy can be utilized for many different kinds of purposes, including online review systems, market research, and social media monitoring can give us important information about the tastes and opinions of our customers.

1.2 Challenges

While creating our sentiment analysis model for Arabic and English movie reviews, we encountered a number of difficulties. The lack of readily available datasets for Arabic movie reviews was one of the major difficulties. A lot of digging had to be made to look for an Arabic movie review dataset because the majority of existing sentiment analysis datasets are in English. At first, we thought we would have to collect the data ourselves, but after extensive searching, we eventually located one. Additionally, the complexity of the morphology and the scarcity of language resources in Arabic present certain particular difficulties that made preprocessing and modeling more difficult. Additionally, the datasets contained reviews with varying lengths.
To solve these challenges, we preprocessed the data by deleting stop words, punctuation, and other unnecessary characters in order to get around these problems. Additionally, we employed padding to make sure that all reviews were the same length and a tokenizer that could accept Arabic text.

1.3 Datasets

In this project, we used two datasets for sentiment analysis of movie reviews in Arabic and English.
The first dataset is the "Arabic Movie Reviews Dataset", which is a publicly available, consisting of 50001 movie reviews in Arabic. The dataset is divided into two classes: positive and negative reviews, with 25001 positive samples and 25000 negative samples. The reviews were collected from various Arabic movie review websites, and they cover a wide range of movie genres.
The second dataset we used is the "IMDb Movie Reviews Dataset", which is a popular dataset for

sentiment analysis of movie reviews in English. It contains 50,000 movie reviews, split into 25,000 for each class. The reviews are labeled as positive or negative, and they were collected from the IMDb website.

2 Data Analysis

The Data retrieved from IMDb did not need further cleaning, we were able to see that the reviews were evenly distributed on both positive and negative classes. Both datasets were preprocessed to remove stop words, punctuation, and other unnecessary characters. In particular, the English dataset was converted to lowercase, with punctuation and stop words removed. It was then tokenized and lemmatized using the NLTK Library [1]. On the other hand, the Arabic dataset was preprocessed by removing diacritics, stop words, punctuation, and English characters. Afterward, tokenization and stemming, using ISRIStemmer [2], were applied.

2.1 Limitations of Data Set

A small dataset of 50k entries for each language causes over-fitting due to less training data. Also, another limitation is that all data is from one source which is IMDb. Another one was for the Arabic dataset which does not cover all Arabic dialects.

3 Methodology

In this chapter, we discuss our whole implementation which involves the preprocessing and different approaches for building the model. We used Google collab to implement the project, we mounted google drive in order to save the datasets and the model so that we do not waste time each time we open a session by re-uploading the datasets or re-training the model.

3.1 Preprocessing

3.1.1 English Preprocessing

We started by preprocessing the English dataset, first of all, we defined a function that takes an input string and converts it to lowercase, then removes numeric values, then tokenizes it using the NLTK library. Next, it lemmatizes the text and finally, it returns the text after doing all these things. We apply this function on the review column of the English dataset and save what is returned in a newly created column called ”preprocessed_text”.

3.1.2 Arabic Preprocessing

Arabic preprocessing is similar to English preprocessing, we defined a function that takes an input string as well however, Arabic text may contain English text by mistake from the keyboard and also it may contain diacritics. Therefore we started by removing diacritics followed by removing English text then we tokenized the text then we stemmed the text and finally we returned the text after editing. Next, we apply this function to the review column on the Arabic dataset and save what is returned in a newly created column called ”preprocessed_text”.

3.2 Model Building

We tried three different approaches to see which one would be more accurate and give better results.

3.2.1 1st Approach

In this approach, we concatenated the English and Arabic datasets together and feed them to the model. We did a train, test split using skearn library [3] on the new concatenated dataset where the feature is the preprocessed text and the label is the sentiment with a test size of 20%. We created an instance of the Tokenizer imported from Keras library[4] class with the parameter num_words set to 5000. This parameter specifies the maximum number of words to keep based on their frequency. Only the most common num_words words will be retained in the tokenized data. Next, We fit the tokenizer on the training data which means it will analyze the text corpus and build its internal vocabulary. Then, we use the tokenizer to convert the sentences in the training data into sequences of integers. Each word in the sentences is replaced by its corresponding index in the tokenizer's vocabulary. Similarly, we converted the test data into sequences of integers as well. Lastly, we mapped the positive sentiments in the labels to value 1 and the negative sentiments to value 0. In summary, Following these steps, we transformed the text data into a numerical representation that can be used as input to a machine learning model. This allows us to leverage various techniques and models for text classification and other natural language processing tasks. Then we padded the train and test data if needed to 100 as the model training accepts only inputs of the same length. Finally, we started building the model which was an LSTM model. First, we created a sequential model object, which allows us to build a deep learning model layer by layer. Second, We added an embedding layer to the model. The embedding layer is responsible for mapping the input tokens (integer-encoded words) to dense vectors of fixed size. We specified it to have 128 dimensions for the embedding vectors. Third, We add an LSTM layer to the model. We set it to have 128 hidden layers. we also set the dropout and recurrent dropout arguments to 0.2 to prevent over-fitting. Fourth, We added another LSTM layer to the model, but this layer has 64 hidden layers. Fifth, We add a dense layer to the model. The dense layer is a fully connected layer where each neuron is connected to every neuron in the previous layer. We set it to have 64 neurons and the activation function to Rectified Linear Unit (ReLU). Lastly, we add the output layer to the model. The output layer has a single neuron and uses the sigmoid activation function. This is suitable for binary classification tasks, where the output represents the probability of the positive class. Finally, we fit the model and trained it for 5 epochs which resulted in an accuracy of 91% and val_accuracy of 84% which means there is a bit of over-fitting.

3.2.2 2nd Approach

In this approach, we tried to do machine translation to have a less complex model with better alignment between the two languages. Machine translation is the task of automatically translating text or speech from one language to another using computational models and algorithms. It aims to bridge the language barrier by leveraging large amounts of bilingual or multilingual data to learn patterns and mappings between languages. We used the Transformers [5] library in Python. We started by creating a pipeline instance, the pipeline function is a convenient high-level API that allows you to perform various natural language processing tasks using pre-trained models. It provides a simple interface for common NLP tasks such as text classification, named entity recognition, question answering, sentiment analysis, and machine translation, so we would use it for machine translation in this case. We initialized it to perform Arabic to English translation and fed the Arabic dataset to it. However, we found that it takes too much time to translate and after reading and searching more we found that the translation is not always accurate so we decided not to waste our time and move on to the third approach.

3.2.3 3rd Approach

The final approach, which was using FastText embeddings [6] [7] [8] is another approach for building the model. FastText is a library for efficient learning of word representations and sentence classification developed by Facebook AI Research. It is particularly useful for text classification tasks as it captures not only word-level information but also subword-level information, which can be beneficial for handling out-of-vocabulary words and capturing semantic similarities. To preprocess the English text data, we employed the following steps. First, we utilized the Keras Tokenizer class, which enables us to convert text into sequences of integers. We initialized the tokenizer with a maximum vocabulary size of 5000, and we then fit the tokenizer on the preprocessed text data from the English dataset. This process updates the tokenizer's internal vocabulary based on the text data, enabling it to map words to unique integer indices. Next, we used the fitted tokenizer to convert the preprocessed text data into sequences of integers. Each word in the text was replaced with its corresponding integer index from the tokenizer's vocabulary. These sequences of integers serve as input for our neural network model. To ensure that all sequences have a fixed length, which is necessary for feeding them into a neural network, we padded the sequences using the pad_sequences function from Keras. In our case, we set the maximum sequence length to 100. If a sequence was shorter than 100, zeros were added at the beginning to make it equal to the maximum length. If a sequence was longer, it was truncated from the end. To enhance the representation of words in our model, we incorporated pre-trained FastText word embeddings for the English language. These embeddings capture semantic and syntactic information about words. We loaded the embeddings using the Gensim [9] library's KeyedVectors class from the file wiki.en.vec. To construct the embedding matrix, we considered the words in our tokenizer's vocabulary and matched them with their corresponding pre-trained embeddings. We initialized an empty matrix of dimensions (the number of English words which was 5000, 300) and populated each row with the word embeddings. If a word was not found in the pre-trained model's vocabulary, we assigned a random embedding vector using a normal distribution with a mean of 0 and standard deviation of 0.25 as a fallback option for out-of-vocabulary words. By performing these preprocessing steps and incorporating pre-trained FastText embeddings, we aimed to enhance the quality of our input data and provide meaningful representations for the words in our model. These steps collectively contribute to the overall effectiveness and performance of our English text processing pipeline. The same was done for the Arabic dataset but using wiki.ar.vec instead wiki.en.vec. Now moving onto the model building, the model architecture we created has a number of layers and components that work together to analyse and categorise text input with regard to sentiment in both English and Arabic. The input layers for the English and Arabic text data are first defined. The input sequences, which are collections of integer indices representing words in the different languages, are defined by these layers. Then, we include embedding layers for inputs in both English and Arabic. These layers convert the input word indices into word embeddings, which are dense vector representations. We used the FastText embeddings that we created with the LSTM layers, for both English and Arabic inputs after the embedding layers. Long-range relationships in sequential data can be modeled using LSTMs. The model is able to identify complex patterns and correlations in the data by using LSTMs to analyse the embedded representations of the input texts. We employ a concatenation layer to merge the data retrieved from the LSTM layers. The outputs of the English and Arabic LSTM layers are combined in this layer to provide a single representation that includes features from both languages.The final layer was the output layer, which consisted of a single unit with a sigmoid activation function. This unit produces a probability-like output between 0 and 1, representing the predicted sentiment label. The sigmoid activation function allows the model to assign a sentiment score to each input, indicating the probability of it being positive or negative. We did not include two LSTM layers and two output layers as the first approach in order to save time since we were short on time, however, we trained the model for ten epochs instead of five which resulted in an accuracy of 99% and val_accuracy of 85%; we can see there is over-fitting here as the first approach but in that case, the accuracy and val_accuracy are much higher. Since we had two different input layers for the English and Arabic inputs, we used langid [10] library on the input text and if it is English we would input it in the first layer and set the second layer with zeroes and the opposite was done if the input is Arabic.

We found that in our case the first approach showed better results. However, the third approach makes more sense since it gives a better understanding of the input in the case of Arabic or English. So we decided that the first approach would work better in our case due to the limitation of the small dataset but in the real world where we would have a large dataset, the third approach would give much more promising results.

4 Results

We then experimented with the first and third approaches to figure out the better approach which leads to a better overall result. We did not experiment with the second approach as stated in the methodology chapter, we did not continue with it since it took too much time.

4.1 Testing the first approach

For the first approach, after training, we got an accuracy of 84.38%, a Precision of 80.29%, a Recall of 90.99%, and an F1-Score of 85.30% Table 1. As seen in Figure 1 the receiver operating characteristic (ROC) curve has produced an area under the curve (AUC) of 0.93.
As a use case, we tested the model on two confusing samples of data that it had never seen before. the first sentence was "I really enjoyed the Marvel works I am sad that I have not watched the other movies" and the second sentence was "The movie was terribly amazing". Their Arabic equivalent was also used,

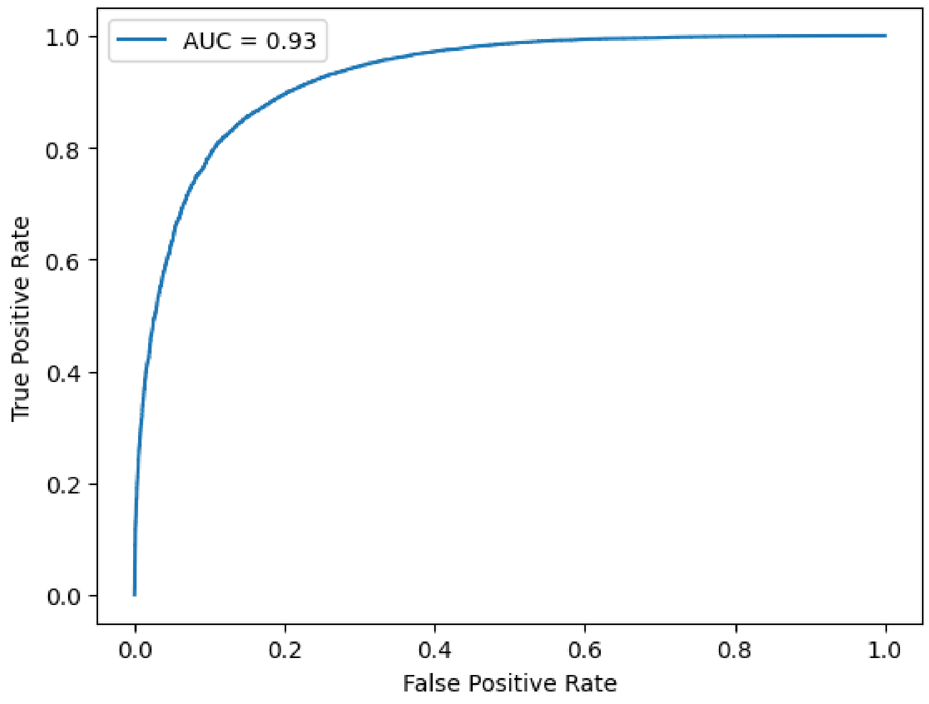

Figure 1: receiver operating characteristic (ROC) curve approach 1

”كان الفيلم مذهلاً بشكل رهيب” and ”لقد استمتعت حقًا بأعمال مارلفل ، وأنا حزين لأنني لم أشاهد الأفلام الأخرى” respectively. They were all classified correctly as positive using this approach. Another use case was used as well which was: ”The movie had some good scenes, however in overall I did not like it” and its Arabic equivalent ”كان للفيلم بعض المشاهد الجيدة ، ولكن بشكل عام لم يعجبني”, negative sentiment was classified in these two cases.
A confusion matrix is usually used to provide a detailed breakdown structure of the model's performance by showing the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These values are essential for understanding the classification results. So we also printed the values of the confusion matrix By examining the values in the confusion matrix, we can evaluate the model's performance in terms of accuracy, precision, recall, and F1-score. Additionally, the confusion matrix allows us to gain insights into the specific types of errors the model is making, such as false positives or false negatives Table 2.

Table 1: Results

Metric	Value
Accuracy	0.85
Precision	0.82
Recall	0.89
F1-Score	0.86

Table 2: Confusion Matrix

Predicted/Actual	Positive	Negative
Positive	8581	1452
Negative	1539	8429

4.2 Testing the third approach

The same testing methods were used in the third approach. It had higher accuracy of 99% and val_accuracy of 86% while training. This over-fitting caused worse prediction results with less accuracy, recall, and f1-score and even not accurate results when predicting on input texts. The model had an accuracy of 50%, precision of 50%, recall of 48%, and F1-Score of 48% Table 3. As seen in Figure 2 the receiver operating characteristic (ROC) curve has produced an area under the curve (AUC) of 0.5.
As a use case, we tested the model on two confusing samples of data that it had never seen before. the first sentence was "I really enjoyed the Marvel works I am sad that I have not watched the other movies" and the second sentence was "The movie was terribly amazing". Their Arabic equivalent was also used, "لقد استمتعت حقًا بأعمال مارلفل ، وأنا حزين لأنني لم أشاهد الأفلام الأخرى" and "كان الفيلم مذهلاً بشكل رهيب" respectively. For the first one, it predicted a positive which is correct. However, for the second one, it predicted a negative which is incorrect.
We also used the confusion matrix to double-check the results Table 4.

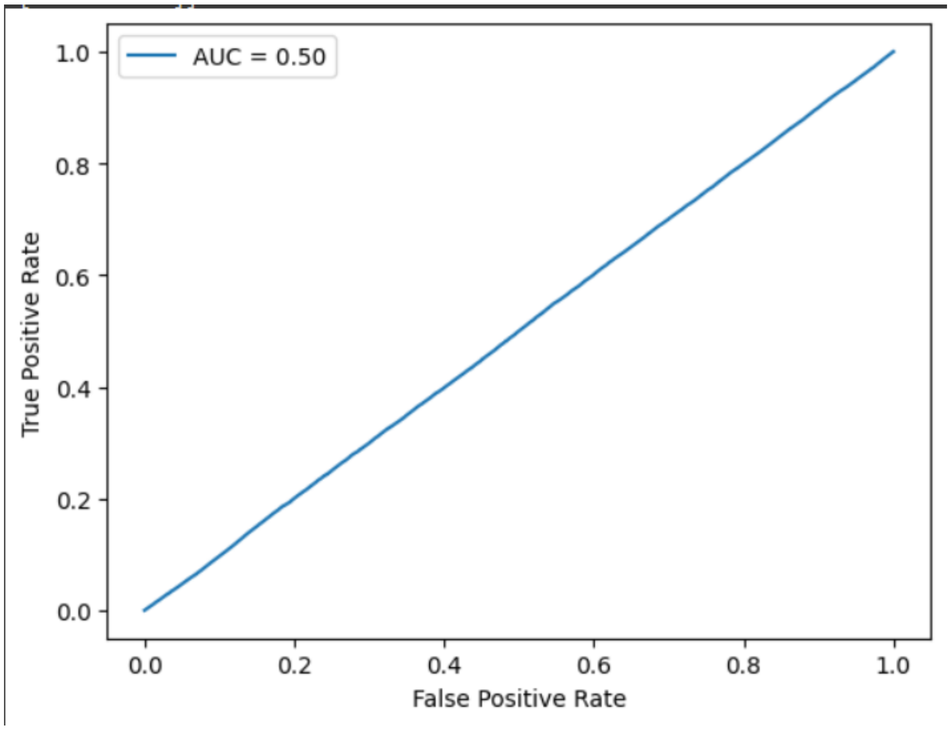

Figure 2: receiver operating characteristic (ROC) curve approach 3

Table 3: Results

Metric	Value
Accuracy	0.5
Precision	0.5
Recall	0.48
F1-Score	0.48

Table 4: Confusion Matrix

Predicted/Actual	Positive	Negative
Positive	26497	23503
Negative	26569	23431

We can see in the end that although the third approach was supposed to be better since it gives

the model a better understanding in different languages, it gave much worse results but this issue is due to the over-fitting because of the small datasets. If the dataset is big enough, the third approach would give much better results and maybe even better than the first approach. Also, the first approach gave better results since we concatenated the two datasets so it had 100k entries which is double the entries for the third approach since the third approach was trained on each dataset alone.
The results ensured the limitation provided in the beginning which is the small dataset would cause over-fitting.

5 Conclusion

In conclusion, the goal of this research was to create a Natural Language based sentiment analysis system for movie reviews in Arabic and English. It was developed to overcome the drawback of existing sentiment analysis algorithms, which were made primarily for English text and might not function well with texts written in Arabic. The study encountered difficulties because of the dearth of Arabic movie review statistics and the difficulty of finding Arabic language materials.
We used two datasets—the Arabic Movie Reviews Dataset and the IMDb Movie Reviews Dataset—to address these issues. In order to prepare the datasets for data analysis, stop words, punctuation, and other unwanted characters have to be eliminated. The datasets have some drawbacks, such as their small size and the fact that they were from a single platform, which led to over-fitting.
The various approaches used to create the sentiment analysis model were described in the methodology section. The first method entailed joining the English and Arabic datasets together, then using the combined data to train a Long Short-Term Memory (LSTM) neural network. This method had an 89 percent recall rate but also showed significant over-fitting characteristics. In the second approach, we attempted to use machine translation for alignment between the two languages but this was abandoned due to translation errors and time restrictions. Finally, the third strategy comprised employing FastText embeddings and LSTM layers for both English and Arabic inputs. This method's recall rate of 48% is a clear indicator of over-fitting.
The difference between the first and third approaches is how a model can well understand complex input without the model being too complex; The first model was too complex but the second one is not complex, in fact, it is using already trained embeddings. Overall, the project showed how natural language processing and deep learning techniques might be used to construct a successful sentiment analysis system for Arabic and English movie reviews. The results could advance the field of natural language processing and give movie producers, directors, and viewers insightful information. To enhance the performance of the model, it is crucial to take into account the datasets' constraints and look into the possibilities of increasing their size and including a variety of sources.

6 Future Work

Based on the approaches tested in this paper their outputs and limitations, it is important to tackle the root of the problems in future work which is the data at hand. Firstly, both the Arabic and English data should increase to a great extent, it is obvious that 100k entries of data are not enough especially when split as two separate datasets at training time. More movie reviews should be scraped and labeled. It is also important to take reviews from multiple platforms such as rotten tomatoes and its counterparts. Furthermore, the second approach here should be tested to its full extent. This could be achieved by either having a proper translation tool that is both fast and reliable or in the worst-case scenario having the data manually translated or reviewed by professionals. It will be important to integrate such a translation tool into the model for testing novel data given in the Arabic Language.

References

[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[2] Mochamad Gilang Syarief, Opik Taupik Kurahman, Arief Fatchul Huda, and Wahyudin

Darmalaksana. Improving arabic stemmer: Isri stemmer. In *2019 IEEE 5th International Conference on Wireless and Telematics (ICWT)*, pages 1–4, 2019.

[3] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[4] Francois Chollet et al. Keras, 2015.

[5] family-names: Wolf given-names: Thomas family-names: Debut given-names: Lysandre family-names: Sanh given-names: Victor family-names: Chaumond given-names: Julien family-names: Delangue given-names: Clement family-names: Moi given-names: Anthony family-names: Cistac given-names: Perric family-names: Ma given-names: Clara family-names: Jernite given-names: Yacine family-names: Plu given-names: Julien family-names: Xu given-names: Canwen family-names: "Le Scao" given-names: Teven family-names: Gugger given-names: Sylvain family-names: Drame given-names: Mariama family-names: Lhoest given-names: Quentin family-names: Rush given-names: "Alexander M.". Transformers: State-of-the-art natural language processing, 2020.

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[7] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[8] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[9] Radim Rehurek and Petr Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2), 2011.

[10] Marco Lui and Timothy Baldwin. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea, July 2012. Association for Computational Linguistics.